Exploring a New Transport Protocol for Vehicular Networks

Augustine Ezenwigbo*, Vishnu Vardhan Paranthaman*, Ramona Trestian*, Glenford Mapp* and Fragkiskos Sardis†
*Faculty of Science and Technology
Middlesex University, London SE12 9HY
Email: A.Ezenwigbo, V.Paranthaman, R.Trestian, G.Mapp@mdx.ac.uk
†Centre for Telecommunications Research, Department of Informatics
King's College London, London, UK
Email: fragkiskos.sardis@kcl.ac.uk

***Abstract*—The Future Internet will be very different from the current Internet. In particular, support for new networks such as vehicular networks, will be a key part of the new environment. Applications running on these networks will require low latency and high bandwidth, which must be provided in a highly mobile environment. The goal of this paper is to look at these issues as they have been addressed in the design and development of the Simple Lightweight Transport Protocol (SLTP) to support vehicular networking. The functions and workings of the protocol are examined in this paper as well as the ecosystem that is needed to provide low latency. A detailed set of preliminary results are presented and compared with a standard TCP implementation. SLTP was also ported to the Roadside Units of a Vehicle Ad-Hoc Network and results are presented for moving data to and from the Roadside Units. This work highlights the need for the Future Internet to place more resources at the edge of the core network to provide support for low latency in vehicular environments.**

***Index Terms*—Vehicular Network, Transport Protocol, SLTP**

I. Introduction

The Future Internet will have to support new networks. This is because the emergence of Connected and Autonomous Vehicles (CAVs) will make vehicles first class communication entities. In addition, CAVs will have to provide a heterogeneous environment where many networking technologies such as IEEE 802.11p, 5G, WiFi and Bluetooth Low Power (BLE) will be supported. However, due to the high mobility in vehicular networks, the provision of low latency and high bandwidth communication will be essential for applications running in this new environment. These requirements are difficult to fulfil using the current IP framework and so a new approach is required. This paper looks at the issues involved in building low latency transport protocols for vehicular environments.

For years we have used the two main transport protocols designed for communication over the Internet: the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP has been used to provide reliable stream like communication while UDP is used to provide an unreliable datagram service. Going forward however, there is a need to provide reliable service using one protocol which has both the reliablity of TCP and the low latency of UDP. In addition, the emergence of Cloud and Edge computing systems mean that more work is being done in the local area. Mobile services on Edge clouds will also be used to lower the latency between clients and servers as users move around and hence it is necessary to ensure optimal communication in such environments [1].

In the application space, more applications are aware of Quality of Service (QoS) issues and how they affect the Quality of Experience (QoE) as perceived by the user. Thus for most applications, there is a desire to move away from kernel based protocols such as standard TCP to user level transport systems where the transport protocol runs with the application in user space. This approach is also supported by the development of Software Defined Network (SDN) and Cloud Computing systems which allow a much finer control of network flows and computing facilities.

A new transport protocol called the Simple Lightweight Transport Protocol (SLTP) has been designed to explore these issues. This paper examines SLTP structures, functions, and the overall ecosystem. The results show that this approach performs well in high capacity environments, and hence, points to the need for Edge systems to support vehicular environments. The rest of the paper is structured as follows: Section 2 looks at emerging technologies for vehcular networks and their requirements at the transport level. Section 3 details the new transport protocol (SLTP) while Section 4 shows how the protocol manages connections. Section 5 looks at the Vehicular Ad-Hoc Network (VANET) Testbed for which the SLTP protocol was developed. Section 6 gives preliminary results. Section 7 looks at related work and Section 8 concludes the paper.

II. Vehicular Networks

The rapid growth in the number of vehicles on the roads has created a plethora of challenges for road traffic management authorities such as traffic congestion, increasing number of accidents, air pollution, etc. Over the last decade, significant research efforts from both the automotive industry and academia have been undertaken to accelerate the deployment of the Wireless Access in Vehicular Environments (WAVE) standard based on a Dedicated Short Range Communication

(DSRC) among moving vehicles (Vehicle-to-Vehicle, V2V) and roadside infrastructure (Vehicle-to-Infrastructure, V2I). This network is called a VANET (also nown as ETSI-G5) network and is characterised by high node speed, rapidly changing topologies, and short connection lifetimes as shown in Figure 1. VANETs are realised by the deployment of Roadside Units (RSUs) located along the transport infrastructure and Onboard Units (OBUs) in the vehicles or worn by pedestrians or cyclists.

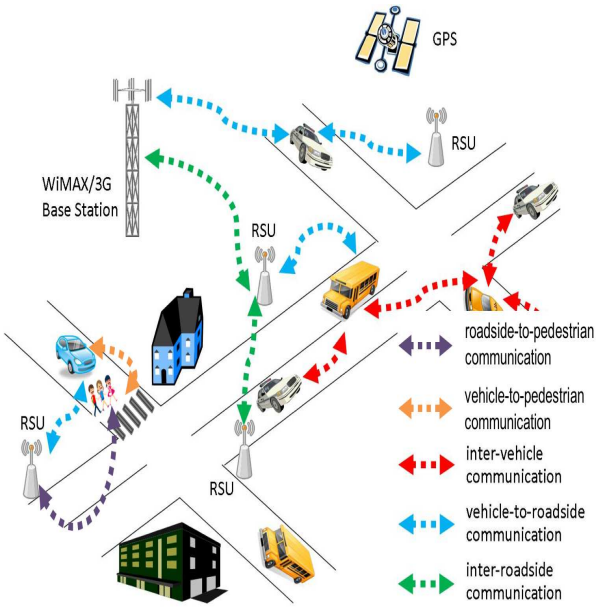

Fig. 1. VANET Communication

A. The challenge of providing seamless connectivity in vehicular environments

Because of the high mobility, support for seamless communication in vehicular networks face a number of challenges:

- **Short connectivity times in individual networks**: Because of the high speed of the vehicle, the actual connection time of the vehicle to the Internet infrastructure is very short (in the order of tens of seconds for slow moving vehicles). This means that low latency communication is required at the transport level so that servers in the Internet can better communicate with fast moving vehicles.
- **Frequently handovers**: Because of the short communication time, there are frequent handovers between OBUs and RSUs as vehicles move around. Hence at the transport level, it is necessary to support fast connection establishment and termintaion times.
- **Support for multicast communications**: In order to support quick handovers as well as streaming applicatrions, using multicast communications where an OBU could be connected to more than one RSU at the same time will enable a smoother transition between networks.
- **QoS support**: As a connection will have a short duration time, it is necessary to be able to use the entire bandwidth of connection once it has been established.
- **Minimum Transport Signature**: In order to ensure that most of the communication bandwidth is available for applications, the number of packets sent at the transport level needs to be kept to a minimum. This means that only data packets that are missing on a reliable connection should be retransmitted. In addition, acknowledgements need to be used in a more directed way in order to minimize the retransmission of acknowledgements as well as data.

These requirements have been a challenge when using standard protocols such as TCP and hence it was decided to explore a new approach to see if the a new transport protoocol could be developed to meet such requiremennts.

B. Implementation Issues

This section explores the implementation issues involved in developing a transport protocol to support vehicular environments. These include:

- **Running Efficiently in User Space:** Since low latency is required in these environments, it is important that there is minimum crosstalk between applications. This points to the need to have the protocol run in user space because running transport protocols in the kernel results in a huge amount of crosstalk for all applications. This means, for example, a reliable fast video stream could be affected by activities from other applications. Traditionally, running in user space meant that there would be a limited amount of CPU cycles to run transport protocols and hence, such systems were inherently slower than kernel-based protocols such as TCP. However, because of the development and proliferation of multiprocessor architectures, there is now a large amount of idle user space CPU cyles. Hence, support for user space protocol processing is being actively pursued by several companies including TCP Onload [2] and the Data Plane Development Kit (DPDK) initiatives [3]. However, these efforts use very sophisticated hardware engines rather than designing the protocol itself for low-latency.
- **Protocol ecosystem:** In addition, to run efficiently in user space, key issues must be addressed in order to develop a highly efficient ecosystem. This includes memory management, timers, packet and buffer management and providing up-calls from the protocol to the application or server. Providing such facilities is important for efficient implementation of the proposed protocol. However, these factors are also very dependent on the hardware, software, and operating system that are being used.
- **An Event based interface between the transport protocol and the application:** Most transport systems use PUSH-PULL mechanisms developed by the traditional socket layer libraries where senders transmit or PUSH data towards the client while receivers retrieve or PULL

the data from the underlying socket for the connection. However, by monitoring changes in the protocol state as well as packet delivery, it is possible to use an event based mechanism in which clients and servers simply response to these events. This new interface can make overall communications for servers very efficient.

III. Simple Lightweight Transport Protocol(SLTP)

In this section we look at the core features of the SLTP protocol. The motivation for designing SLTP came from the need to support research into services using Cloud based environments [4] as well as to provide low latency and tunable support at the transport level in vehicular networks.

A. The SLTP Header

Figure 2 shows the Diagram of the SLTP while Table I shows the length of the individual fields.

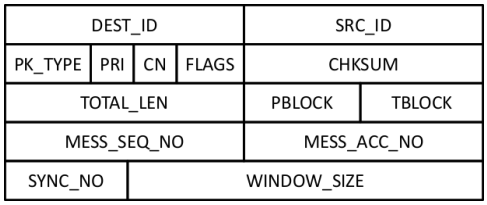

Fig. 2. The Structure of SLTP Header (Total Size is 20 bytes)

TABLE I
The fields of SLTP and their functions

FIELD	BITS	FUNCTION
DEST_ID	16	Connection_Id at the remote end
SRC_ID	16	Connection_Id at the local end
PK_TYPE	4	Type of packet
PRI	2	Priority of the packet
CN	2	Congestion Notification Indication
FLAGS	8	Indicates actions needed to process the packet
CHKSUM	16	Uses the TCP Checksum
TOTAL_LEN	16	Total length of the packet
PBLOCK	8	Current block or fragment
TBLOCK	8	Total number of blocks in the message
MESS_SEQ_NO	16	Sequence number of the last message sent
MESS_ACC_NO	16	Sequence number of the last message received
SYNC_NO	12	Random number to prevent replay attacks
WINDOW_SIZE	20	The Receive Window Size

B. SLTP Packet Types

SLTP supports a number of packet types as shown in Table II.

C. SLTP Flags

SLTP FLAGS comprises a field containing 8 bits. Their functions are detailed in Table III.

TABLE II
Packet types and their functions

PACKET_TYPE	FUNCTION
START	First packet transmitted on a connection
REJECT	Signals that the connection request has been rejected
DATA	Data packet
ACK	Acknowledgement (ACK) packet
NACK	Used for selective retrnsmission
END	Used to close a connection
FIN	Final packet sent
ECHO	Used to measure RTT
ECHO_1	First back-to-back packet
ECHO_2	Second back-to-back packet
STATUS	Used to maintain flow control
IDLE	Sent when there is no data to send
CWIN	Used to change the window size
JOIN	Join a multicast connection
LEAVE	Leave a multicast connecion

TABLE III
Flags and their functions

BIT	NAME	FUNCTION
0	W_VAL	Window-Size is valid
1	ST_CKS	Checksum this packet
2	ST_RTR	Retransmission is permitted
3	ST_RET	Indicates a retransmitted packet
4	REMOTE_RESET	Connection reset by the other side
5	REPLY_REQ	A reply is requested
6	REPLY	Reply to a previous request
7	EOM	Last message was correctly received

D. Support for new flow control mechanism based on QoS

Because of the need for low latency, SLTP uses a new flow control mechanism which measures the latency, bandwidth and burst at the start of the connection. In this approach, two packets of a given size are sent back-to-back and the round-trip times of each packet is measured as well as the time-difference, between the packet replies. In SLTP, ECHO_1 and ECHO_2 packet types are used to perform this test. A speical timer is used to ensure that this test is successfully completed. This test is repeated once NACK packets are received to ensure that the connection adjusts to changes in the overall networking conditions.

The latency of the connection is measured by using the average RTT of ECHO_1 and ECHO_2. We calculate the bandwidth by dividing the bytes transferred by the time taken to transfer them. The burst size is measured by taking the average RTT and dividing by the time difference between the replies to ECHO_1 and ECHO_2; this is then multiplied by the size of the ECHO packets. This value represents the amount of bytes needed to keep the connection pipe full and thus is the burst size of the connection. Hence, we set the maximum number of unacknowledged bytes to this value. Sending data after this value has been reached may result in significant data loss as the pipe is already full. This approach is similar to the packet-pair approach which has been used to estimate the bandwidth of a connection. However, this is being used in SLTP to allow the full bandwidth of the connection to be

used as soon as the transport connection is established. Hence there is no need for the slow start and congestion avoidance algorithms that are employed in TCP.

E. An event-based interface for SLTP

In addition to the traditional socket interface for sending and receiving data, SLTP has also implemented an event interface based on the following events:

- A new connection is started
- A new message has been delivered to the application
- The remote side has closed the connection
- The connection is closed at both ends and hence resources can be reallocated
- There is a security violation detected on the connection
- There has been a network reset

The event interface is shown in Fig.3. In addition, applications can supply up-calls to functions that should be called when messaages are received. This minimises the latency with which servers can deal with new requests. Support for handovers in SLTP is done using the ability to change the window size, similar to the zero window size mechanism in TCP. However, in SLTP this can be done by the application itself. So when the application is notified about an impending handover, it closes its window i.e., the window size is set to zero, and sends a change window packet to the sender indicating the reason for the new value. If the new value is zero, the sender is obliged to poll the receiver with STATUS packets until the window is opened again.

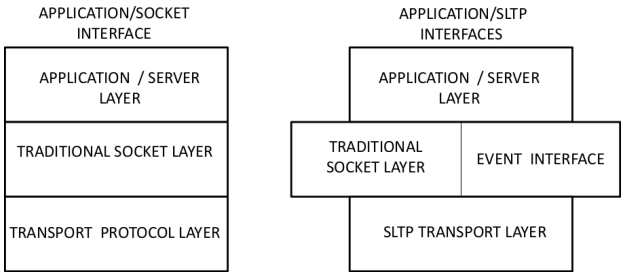

Fig. 3. Traditional vs SLTP Interface

F. Protocol Ecosystem in SLTP

In order to ensure low latency, other functions need to be supported. One of them is memory management because the protocol must not be slowed down due to a lack of kernel memory resources. In SLTP, each application manages its own memory heap (around 20MBs) on behalf of the protocol. Hence, buffers which hold incoming packets, connection and timer structures are all managed internally by SLTP. In addition, timers are been implemented efficiently using a separate timer thread.

G. Minimizing packet retransmission in SLTP

In SLTP, missing packets result in NACK packets being sent back to the sender indicating which packets are missing. Hence, only these packets are retransmitted. In addition, in SLTP acknowledgements are controlled by the sender and so requests for acknowledgements are used to detect problems in the connection before packets are retransmitted. This ensures more intelligent retransmission policies.

IV. Operation of SLTP

Fig.4 shows the state diagram for SLTP. Connections are started by sending a START packet to the receiver with the REPLY_REQUESTED bit set in the flags field of the SLTP header. The receive, on accepting the connection, replies with a START packet with the REPLY bit set. Both ends go to the CONNECTED state. If either end closes the local connection, it goes to the END_LOCAL state. If, however, the other side closes the connection first, the connection goes to the END_REMOTE state. In the END_LOCAL state, once all the packets have been sent and acknowledged, an END packet with the REPLY_REQUESTED bit set is sent. If the other side also wishes to close the connection, it sends an END packet with the REPLY bit set and goes to the TIMED_CLOSE state. Finally, a FIN packet is sent to close the connection and deallocate resources.

Hence SLTP uses a two-way connection establishment mechanism which ensures quick connection setup. To end a connection, two END packets are used but a FIN packet is added to quickly bring down the connection at both ends. Thus in SLTP, there is no need to have long delays to completely end a connection.

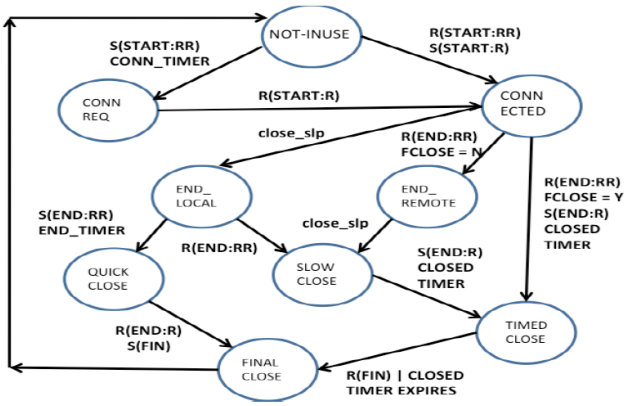

Fig. 4. SLTP Connection

V. VANET Testbed

The Department for Transport (DfT) and Middlesex University have built a Connected Vehicle Testbed using VANET technology at its Hendon Campus. The testbed has seven RSUs: four on the university campus buildings and three along the A41 road behind the university. RSUs and OBUs were manufactured by Lear Corporation in compliance with the IEEE 802.11p (WAVE) standard specifications and the maximum output power used was 200mW or +23dBm. The operating frequency of the RSUs is 5.9 GHz and the channel being used to send broadcasts is CH172. Four RSUs have been deployed on top of the Hatchcroft building, Williams

building, Sheppard library building and Grove building to cover the roads around the campus and to support the movement of pedestrians within the campus, hence enabling the development of Vehicle-to-Pedestrian (V2P) applications. The coverage map of the deployed testbed is as shown in Fig.5 [5].

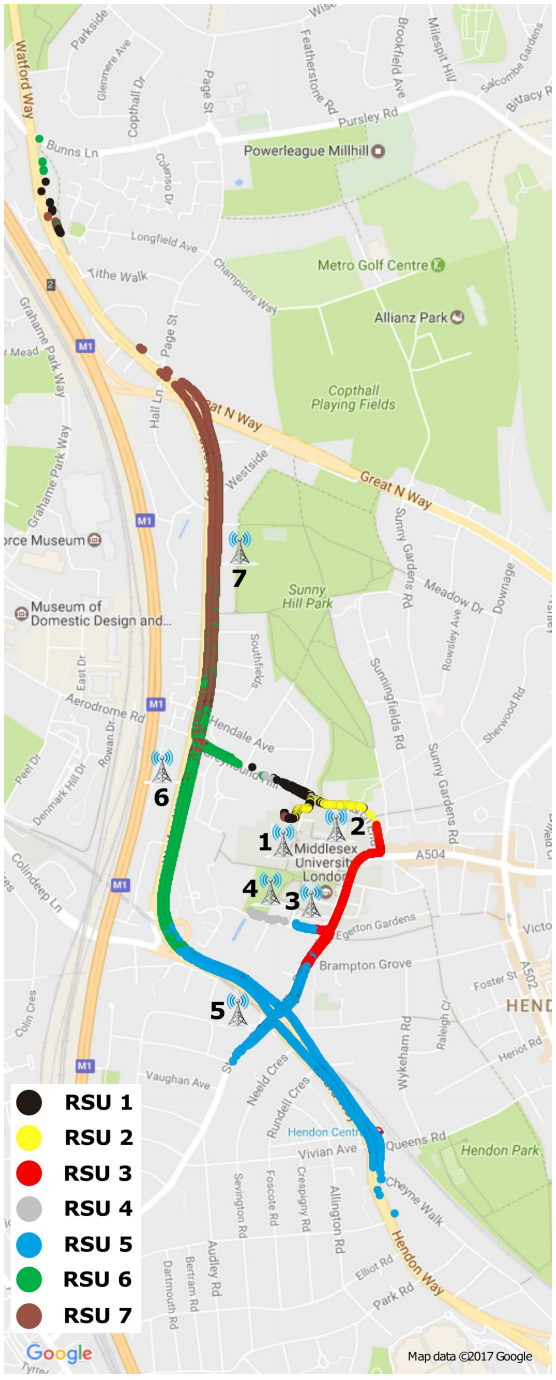

Fig. 5. Coverage Map.

Fig. 5 displays the trial data for a car driving on the roads around the Middlesex University campus to map the coverage which was collected on 17th May 2017. The coverage map shows the individual coverage achieved by the RSUs located on each building and the RSUs located along the A41 road with different colour dots.

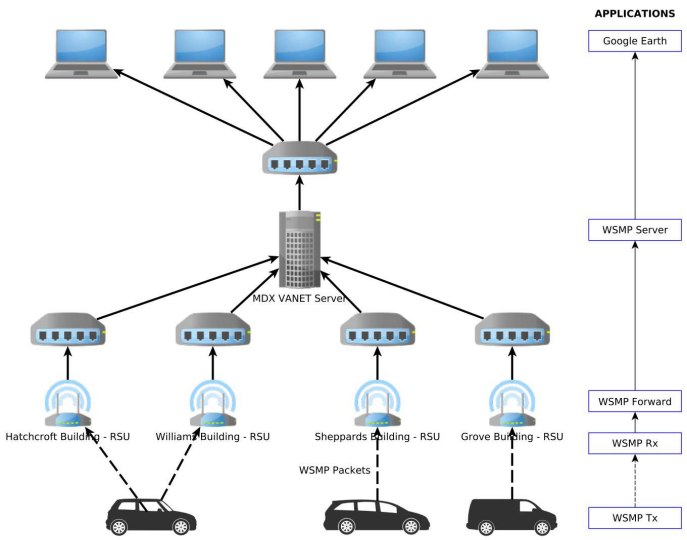

Fig. 6. Network Diagram.

Figure 6 shows the network diagram of the MDX VANET Testbed and the applications running on their respective devices.

VI. Preliminary Results for SLTP

Our first set of results will look at SLTP running on a normal network consisting of two PCs connected together via a one Gigabit/s Ethernet system. Packets of different sizes were sent and the time taken to receive them back at the sender was measured. This gives us a direct measurement of protocol performance. Since SLTP runs over UDP, the size of a single SLTP packet can be up to (64KBs - 8 bytes, the size of the UDP header). However, each SLTP packet had a maximum data size of 1452 btyes to ensure a fair comparison with TCP. Though SLTP supports a window size of 1 MB, it was decided to use a window size of 144 KBs to ensure that receive buffers were not overrun.

A. Results for traditional networking environment

We performed our benchmarks by using two PCs equipped with the following hardware:

- Processor: Intel(R) Core(TM) i5-3770 CPU (4 cores).
- RAM: Both PC with 16GB DDR3
- Storage: Both PC with 320GB HDD
- Network: 1 Gigabit Ethernet University Network
- OS Type: Fedora 25 64-bit

These results shown in Fig.7, indicate that SLTP can perform better than TCP under the conditions outlined. This is primarily due to the fact that these systems have a lot of spare resources including CPU and memory cycles. SLTP running

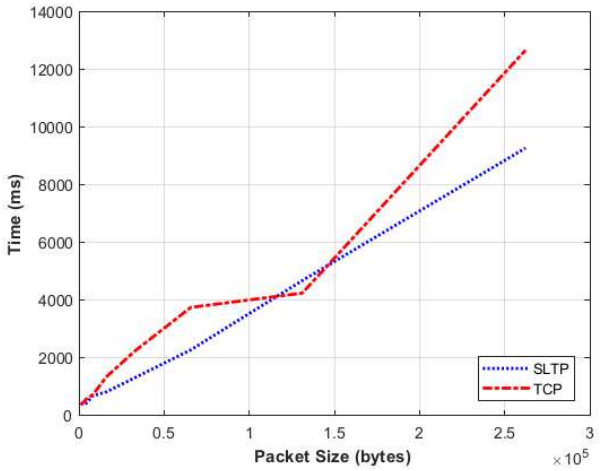

Fig. 7. PC to PC - Network Time: SLTP vs TCP

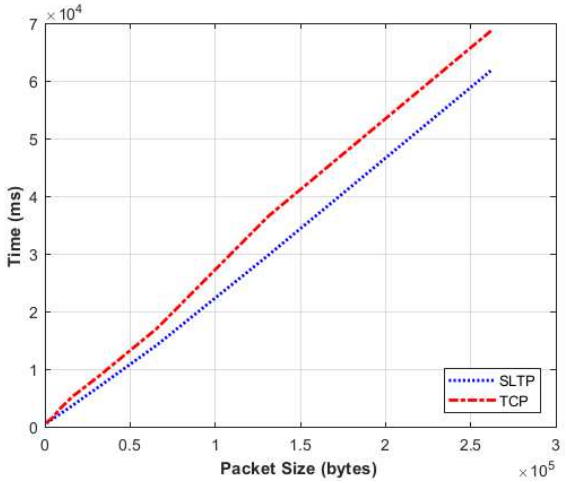

Fig. 8. RSU to VANET Server - Network Time: SLTP vs TCP

in user space can take full advantage of these resources to outperform TCP at higher buffer sizes.

B. VANET server and RSU

In this subsection, we present the results for RSU to VANET communications. We performed our benchmarks by using the following hardware: The specs for the VANET Server are given below,

- Processor: Intel (R) Xeon (R) CPU E5-2683 v4
- RAM: 32GB
- Storage: 500GB
- Network: 1 Gigabit Ethernet University Network
- OS Type: Debian 3.16.43-2+deb8ul x86_64GNU/Linux

The specs for the RSU are given below. It can be clearly seen that the resources on the RSU are quite limited compared with the VANET server or a modern PC.

- Processor: MIPS 24Kc V7.4 (1 core)
- RAM: 64 MB SDRAM (512 Mbits)
- Storage: 16 MB Flash
- Network: 1 Gigabit Ethernet University Network
- OS Type: Linux 2.6.32.27-WAVE_LOCOMATE-200_1.90.0.13_detriot

These results show that in this arrangement, less CPU cycles as well as memory are available on the RSU and hence, the performance of SLTP is not much greater than the standard TCP.

C. RSU to RSU

Finally, the results for RSU to RSU are given in Fig.9. These results show that with two RSUs, there is very limited resources at user level and hence, TCP in the kernel slightly outperforms SLTP as more resources are available in the kernel when running on smaller systems, comapared to running in user space.

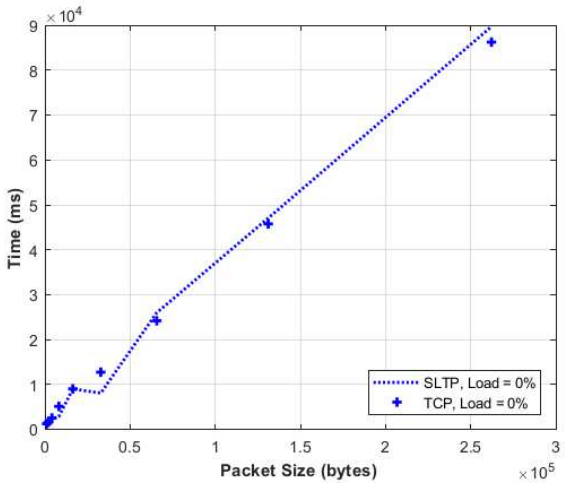

Fig. 9. RSU to RSU - Network Time on Load = 0%: SLTP vs TCP

D. Investigating results for RSUs under different load conditions

Since, SLTP runs in user space, it is important to understand how its performance is affected by different load characteristics of the system. In order to explore this, a flexible hog program was used to remove idle CPU cycles at user level from the system. Hence, we were able to obtain readings with the system being under various loads, including 25%, 50%, 75% and 100%.

Fig.10 shows the time taken to transfer for different buffer sizes under different loads and it clearly shows as the load increases SLTP underperforms TCP as less cycles are available in user space. However, this effect is only significant at very high loads.

The bandwidth results under different loads are shown in Fig.11. There are only significant differences for small packet sizes. However, after around 2KBs, the available bandwidth falls to around 2.5MB/s. This is important for applications

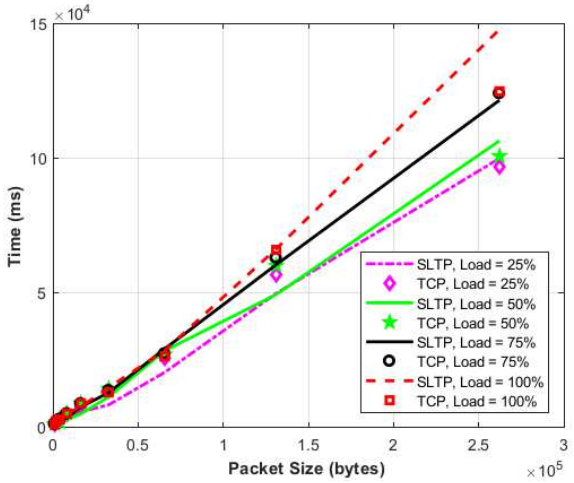

Fig. 10. RSU to RSU - Network Time on Load = 25% to 100%: SLTP vs TCP

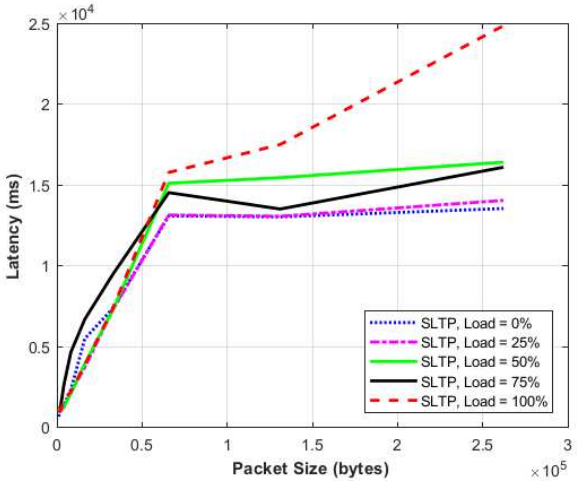

Fig. 12. RSU to RSU - Latency on Load: SLTP

needing large packet transfers sizes such as multimedia applications.

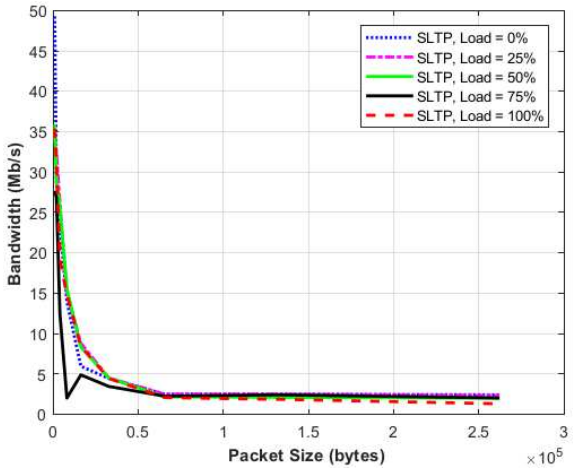

Fig. 11. RSU to RSU - Bandwidth on Load: SLTP

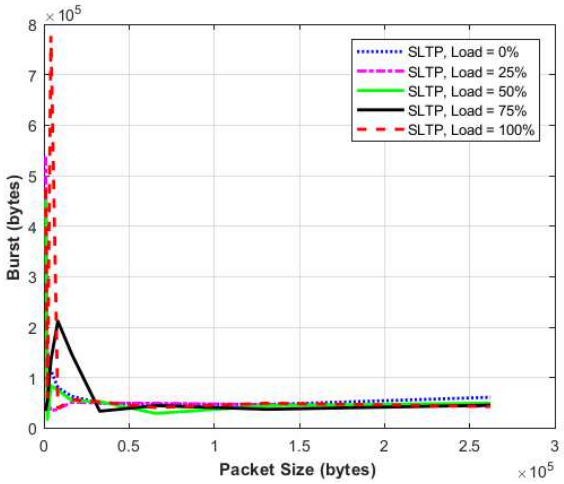

Fig. 13. RSU to RSU - Burst on Load: SLTP

The latency results as measured by SLTP using different packet sizes under different loads are shown in Fig.12. These results indicate that the latency increases with increasing load especially after 50KBs.

Finally, the burst results clearly show that the system is affected by high loads especially for small packets. After around 10KBs, the burst size is severely reduced.

VII. Related Work

In terms of data transport, the Internet has been dominated by the use of TCP and UDP. Other protocols were considered only when these protocols could not be used. For example, the use of the Network Filling System (NFS) required a new protocol running over UDP to reduce the latency [6]. Research into transport protocols that can be used for different types of applications began with the development of the Express Transport Protocols (XTP) [7]. This protocol attempted to optimize new local area networks such as Gigabit Ethernet [8]. It also ran in user space which at that time was regarded as a very limited resource environment. XTP therefore had to support a full range of functions and prototypes making the system complicated to run. However, it did introduce new concepts such as the request and reply bits and sync numbers.

The next big effort in transport protocol design came with research into new networks such as Asynchronous Transfer Mode (ATM). Attempts were made to support multimedia applications running over these networks. The A1 transport protocol was developed for this environment [9]. It also ran in user space and attempted to support the idea of streams having a specified QoS based on a QoS vector which was supplied by the application. It included features such as maximum burst and jitter values as well as the priority of the stream. In addition, attempts were made to develop an efficient TCP

user space implementation. However, these efforts had mixed results and resulted in the development of TCP offload engines being pursued instead [10].

Support for QoS in the Internet was explored using the Intserv [11] and Diffserv QoS [12] [13] models but they have failed to gain universal acceptance. However, recently the Y-Comm architecture has proposed a Hybrid model which is a combination of Intserv in the Peripheral network and Diffserv in the Core network [14]. More support for QoS at the transport level was seen with the development of Explicit Congestion Notification (ECN). This enables the network to explicitly signal to applications about network congestion using two bits in the transport header [15].

The emergence of commonly used multimedia applications has resulted in the need for low latency. For example, the QUIC protocol is an encrypted, multiplexed, low-latency, user space transport protocol with UDP being used as a substrate. It was developed, tested and deployed at Google's front-end servers [16]. The use of UDP allowed QUIC packets to traverse routers and switches. In addition, the user space development facilitated iterative changes at application update time scales. The results show that QUIC reduced the latency of Google Search responses by 8.0% for desktop users and by 3.6% for mobile users, and reduced rebuffer rates of YouTube playbacks by 18.0% for desktop users and 15.3% for mobile users. Therefore, this work has shown the need to move away from traditional TCP systems for specific applications because of the need for low latency. However, the QUIC protocol was designed to support multi-stream communication, such as Web traffic, in which different types of documents are supported over one communication link. The QUIC protocol is therefore much more complicated than SLTP.

VIII. Conclusions

Thie work presented in this paper has examined the development of a new transport protocol to support vehicular envirnoments and is now part of ongoing research in this area [17]. These results are preliminary and more work is being done to look at how applications can make use of the protocol in vehicular environments. In addition, work is being done to compare this work with other transport protocols.

Acknowledgments

The authors would like to thank the Department for Transport (DfT) for funding this project through Transport Technology Research Innovation Grant (T-TRIG) and Central London Testbed Project (CLTP) grants. (MDX Project code: 102105). We would also like to thank Transport for London (TfL) for their support in building this testbed.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[2] Solarlare, "10 Gbps NIC Cards for Sale." [Online]. Available: storage.dpie.com/products/solarflare/

[3] SlideShare, "Understanding DPDK," February 2015. [Online]. Available: https://www.slideshare.net/garyachy/dpdk-44585840

[4] G. Mapp, D. Thakker, and D. Silcott, "The Design of a Storage Architecture for Mobile Heterogeneous Devices," *ICNS2007*, vol. 0, p. 41, 2007.

[5] V. V. Paranthaman, A. Ghosh, G. Mapp, V. Iniovosa, P. Shah, H. X. Nguyen, O. Gemikonakli, and S. Rahman, "Building a prototype vanet testbed to explore communication dynamics in highly mobile environments," in *International Conference on Testbeds and Research Infrastructures*. Springer, 2016, pp. 81–90.

[6] T. Haynes and D. Noveck, "Network file system (nfs) version 4 protocol," *Network*, 2015.

[7] W. T. Strayer, B. J. Dempsey, and A. C. Weaver, *XTP: The Xpress transfer protocol*. Addison Wesley Longman Publishing Co., Inc., 1992.

[8] V. Jacobson, "A high performance tcp/ip implementation," in *Presentation at the NRI Gigabit TCP Workshop*, 1993.

[9] G. Mapp, S. Pope, and A. Hopper, "The design and implementation of a high-speed user-space transport protocol," in *Global Telecommunications Conference, 1997. GLOBECOM'97., IEEE*, vol. 3. IEEE, 1997, pp. 1958–1962.

[10] J. P. Sterbenz, "Protocols for high speed networks: life after atm?" in *Protocols for High Speed Networks IV*. Springer, 1995, pp. 3–18.

[11] S. Shenker, "Specification of guaranteed quality of service," 1997.

[12] K. H. Chan, J. Babiarz, and F. Baker, "Configuration guidelines for diffserv service classes," 2006.

[13] P. Jones and D. Black, "Differentiated services (diffserv) and real-time communication," 2015.

[14] G. E. Mapp, F. Shaikh, D. Cottingham, J. Crowcroft, and J. Baliosian, "Y-comm: a global architecture for heterogeneous networking," in *Proceedings of the 3rd international conference on Wireless internet*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, p. 22.

[15] D. Black, "Relaxing restrictions on explicit congestion notification (ecn) experimentation," Tech. Rep., 2018.

[16] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.

[17] J. Jeong, Y. C. Sheny, J. P. Jeong, T. T. Ohx, J. Jun, and S. H. Son, "Toms: Tcp context migration scheme for efficient data services in vehicular networks," in *Advanced Information Networking and Applications Workshops (WAINA), 2017 31st International Conference on*. IEEE, 2017, pp. 360–364.